*URL http www elsevier nl locate entcs volume html pages*

*Improving Termination of MDG Based* Abstract State Enumeration

*via Term Schematization*

*S Renault and E Cerny*

*D IRO Universit e de Montr eal C P Succ Centre Ville Montr eal H C J Canada*

*Abstract*

*The MDG approach for hardware veri cation was recently proposed to overcome some limitations of traditional ROBDD based methods of automated veri cation One of its weakness however is that it may su er from the non termination of the state enumeration procedure In this paper we exploit an idea recently proposed of using term schematization to address this problem We propose a strategy to direct the introduction of schematized terms in the state enumeration procedure We show that for certain cases of non termination it signi cantly improves over the previously known approaches*

*Introduction*

*The MDG approach for hardware veri cation has been introduced to en hance and overcome some limitations of traditional ROBDD based methods* of automated veri cation It is based on abstract descriptions of state machines ASM which are represented by a class of decision graphs called Multiway Decision Graphs MDGs MDGs can represent relations as well as sets of states and unlike ROBDDs can support the use of abstract data representations More precisely MDGs allow the use of abstract sorts and uninterpreted functions to represent data values and data operations respec tively To limit the range of possible interpretations of the function symbols that denote data operations it is possible to de ne axioms which partially specify their intended semantics These axioms are given by means of condi tional rewriting rules

*Acknowledgments the work was partially supported by NSERC Canada Nortel Coop erative R D grant No CRD The work was carried out using a workstation on loan from the Canadian Microelectronics Corp*

*c Published by Elsevier Science B V Open access under* [*CC BY-NC-ND license*](http://creativecommons.org/licenses/by-nc-nd/3.0/)*.*

*The main strength of the MDG approach is to integrate the combination of* two powerful techniques namely the implicit state enumeration and the use of abstract sorts and uninterpreted function symbols MDGs have been imple mented and many experiments have been carried out showing that they provide a powerful tool for fully automatic combinational veri ca tion state space exploration veri cation of behavioral equivalence of sequen tial circuits Model checking for a rst order temporal logic using MDGs has also been proposed to express both safety and liveness properties

*Its main weakness however is that in many cases it may su er from the* non termination of the state enumeration procedure Di erent solutions have been proposed to solve certain cases of non terminating problems In a solution is given to a class of problems known as processor like circuits It relies upon the generalization of the set of initial states so as to obtain a larger set of reachable states that is representable by a nite MDG Technically the initial state generalization method is done by replacing abstract constants with abstract variables as initial values of some registers An extension of this method has also been proposed that applies to circuits in which the processor like loop does not start necessarily at the initial state but instead at any other control state It can therefore deal with a larger class of circuits but as a counterpart it requires to detect when the generalization is to be performed i e to nd the entry states of the processor like loops State generalization has been used successfully to achieve termination of the abstract state enumeration on several examples Sometimes this achievement is at the cost of a false negative during invariant checking i e the enlarged set of reachable sets may include states that do not correspond to the intended interpretation of the data operations and for which the invariant to verify does not hold

*Another approach for solving non termination is presented in It is based* on retiming and semantics preserving transformations of the circuit to verify It uses generalization of state variables as in the previous approaches but to avoid the loss of information in the generalization step the value of the state variable to be generalized is kept in a new state variable so that it can be recovered later This method requires the user to have a good understanding of the circuit under veri cation and the skills to perform on it the adequate transformations

*In a new approach to state generalization based on term schematizations* has been introduced The authors show how schematization can be incorpo rated into MDG to solve some cases of non termination due to a state variable which produces an in nite sequence of structurally similar terms One of the main objectives of this approach is to reduce as much as possible the amount of information lost during generalization Schematization exploits the recur ring term structure in in nite sequences of terms in our case the successive values taken by a state variable and provides the least common generaliza tion subsuming those terms This allows not only to maintain the possibility

*of using some information provided by a partial interpretation of the sym bols occurring in the schematization and therefore to increase the chance of* achieving termination but also to reduce considerably the risk of producing false negatives The generalization method proposed in requires the user to supply both the state variable s to generalize and the appropriate schemati zation Moreover the generalization is always done on the initial state which might be an obstacle to some simpli cation via rewriting as it is the case in the initial state generalization method In this paper we propose a strat egy which improves over the approach of and which can deal with more complicated cases of non termination In particular our strategy does not require human guidance the introduction and the handling of schematized terms during reachability analysis is performed fully automatically More over generalization is not performed at the initial step so that one can still fully bene t from a partial interpretation of the abstract functions on the ini tial values Another advantage of postponing the generalization is to allow a state variable to be generalized by di erent term schematizations in alterna tive branches of the state exploration Hence our strategy allows to cope with situations where a state variable gives rise to di erent patterns of divergence in alternative branches of the state exploration

*This paper is organized as follows In Section we brie y recall the concept* of Multiway Decision Graphs and its application to the state exploration of Finite State Machines In Section we recall some notions and operations on recurrence terms In Section we propose an extension of MDGs and of the state exploration procedure based on recurrence terms Section is devoted to some examples

# *MDGs and state exploration*

*In this section we brie y present the framework of MDGs For more details we refer the interested readers to*

*Syntax and semantics*

*MDGs are Directed Acyclic Graphs DAGs which represent formulas in a* variant of rst order logic with equality and sorts with a distinction between concrete sorts and abstract sorts This distinction is a syntactic counterpart of the hardware di erence between data path and control Unlike abstract sorts concrete sorts have enumerations which are sets of individual constants MDGs are limited to a class of quanti er free formulas called the Directed Formulas DFs which satisfy several well formedness conditions in particular canonicity see for details

*De nition Directed Formula Let F be a set of function symbols and*

*V a set of variables We denote the set of terms freely generated from F and*

*V by T F V The syntax of a directed formula is then given by the grammar*

|  |  |  |  |
| --- | --- | --- | --- |
| *below* | *Sort*  *Abstract sort* | *S S* | *S j S*  *j j j* |
|  | *Concrete sort* | *S* | *j j j* |
|  | *Generic constant* | *C* | *a j b j c j* |

|  |  |  |
| --- | --- | --- |
| *Concrete* | *constant C* | *a j b j c j j j j* |
| *V ariable Abstract* | *X*  *variable V* | *V j V*  *x j y j z j* |
| *Concrete* | *variable V* | *x j y j z j* |
| *Directed* | *F ormula Disj* | *Conj Disj* |
|  | *Conj* | *Eq Conj j Eq* |
|  | *Eq* | *A C A T F X* |

*j V C*

*j V A A T F X j*

*j*

*The vocabulary consists of generic constants concrete constants abstract* variables concrete variables and function symbols A DF is a formula in disjunctive normal form each disjunct is a conjunction of equations that cor responds to an alternative path Each equation corresponds to a node edge pair in the corresponding path the left hand side resp the right hand side of the equation being the label of the node resp the edge

*The variables which occur as the label of a node are the primary variables while the other variables are the secondary variables*

*A DF D is of type U V i i the set of abstract primary variables of D is* equal to Vabs ii the set of abstract secondary variables of D is a subset of Uabs and iii the set of concrete variables having occurrences in D is a subset of Uconc Vconc

*The basic MDG algorithms which serve in the procedures for combinational veri cation and reachability analysis include the operations of Disjunction Relational product and Pruning by subsumption A detailed description of these algorithms and others like Conditional term rewriting can be found in*

*Since abstract sorts admit in nite interpretations an abstract description will* represent in nite as well as nite state machines FSMs An abstract de scription will determine a machine for every interpretation MDGs are

*well suited to represent both sets of states and transition output relations of* state machines Let P be an MDG of type U V where U contains only ab stract variables Then for a given interpretation P can be used to represent the set

*Set P f j j U P g*

*V*

*where*

*V*

*denotes the set of compatible assignments of the variables in V*

*that is the set of functions that map every variables x V of sort to an* element x of

*Note that although the formulas represented by MDGs are quanti er free it* is useful to think of the secondary variables as being existentially quanti ed which amounts to viewing the formula as having an existential pre x

*MDG based state exploration*

*Abstract state exploration is the kernel of the MDG tools In particular it is* used by the reachability analysis procedure which veri es that after each tran sition step the outputs satisfy a given property This procedure is described by the pseudo code below where D X Y Z FI FT FO is an abstract state machine description in which X Y Z are disjoint sets of variables viz the input state and output variables respectively and FI FT FO are DFs repre senting the set of initial states the transition relation and the output relation respectively

*Proc ReAn D*

*R FI Q FI K loop*

*K K*

*I Fresh X K*

*N RelP fI Q FT g X Y Q PbyS N R*

*if Q then return success*

*R PbyS R Q R Disj R Q*

*end loop*

*end ReAn*

*R represents the set of visited states and Q the current frontier set State* exploration starts from a set of initial states which forms the initial frontier set Then the Relational Product RelP operation computes the MDG N representing the set of states reachable after one transition from the frontier set Q according to the transition relation The F resh operation ensures that all input variables are renamed into new fresh variables at each iteration of the procedure Then the Pruning By Subsumption PbyS algorithm removes by subsumption from the newly reached states N those that are also in R i e that have already been reached in a previous state The states that are not

*removed form the new frontier from which state exploration continues State* exploration terminates when all the states reachable from the frontier states have been visited Informally PbyS performs paths subsumption as follows Let and be two paths from two MDGs of type U V corresponding to the states s and s respectively By de nition is subsumed by if there

*exists a substitution with domain U such that if is obtained by applying*

*to the nodes and edges of then every node edge pair in is a node edge* pair in A complete description of the PbyS algorithm and other operations on MDGs can be found in

*In general because of abstract variables and the uninterpreted nature of func tion symbols the state exploration algorithm may not terminate It may be* caused by a state variable whose successive values grow in nitely according to a certain recursive pattern Consider for instance an abstract description of a conventional microprocessor where a state variable c of abstract sort wordn

*for n bit words with an initial value equal to a generic constant zero of the*

*same sort is incremented via an abstract function symbol s representing the* incrementation by one Then the state exploration would explore the in nite sequence c zero c s zero c s s zero c s s s zero

*In the remainder of the paper we propose and illustrate a solution to this* kind of non termination based on term schematization which improves over the previous proposal by Ait et al

*Recurrence terms*

*Non terminating computations due to in nite sequences of structurally sim ilar terms is a problem which arises in various elds using rst order terms Term schematization has been proposed to solve this problem by using a meta language based on recurrence terms which are capable of nitely representing* in nite sets of terms As an example the recurrence term A s N zero represents the in nite set S fzero s zero s s zero g N is a vari able which ranges over natural numbers and which serves as a counter spec ifying the iteration level The instance of A via the substitution N is s s s s s zero

*Various schematization formalisms based on the same principle of the itera tion of rst order contexts have been proposed which mainly dif fer in the expressive power Dropping restrictions may increase the expressive* power but a ect the complexity and or decidability of algebraic operations in particular uni cation

*In this paper we restrict ourselves to simple forms of recurrence terms the*

*terms as de ned in We now recall the main de nitions and concepts of* this formalism

*Syntax*

*Let F be a nite set of function symbols V be a countable set of rst order variables be a special symbol of arity be a special constant* symbol serving as a place holder and let V be a countable set of variables over natural numbers called degree variables We assume that V V A position u in a term t is either the root position denoted by the empty string

*or a non root position denoted by i ui if t is of the form f t ti tn*

*and ui is a position in ti We denote by t u the term which occurs in t at* position u and by t s u the term obtained by replacing t u by s

*De nition term A term is de ned inductively as follows*

*Every variable in V is a term*

*If f F has arity n and t tn are terms then f t tn is a*

*term*

*If u is a non root position of h h l u is a rst order term and N is a degree variable then h u N l is a term*

*h u N l is called a generator h u represents the recurring pattern and is referred to as the iterated context and l is the non recurring part of the term usually called the base term*

*Semantics*

*A term represents either a rst order term when its degree variables are* all constants or an in nite set of rst order terms when it contains a non instantiated degree variable We denote by H n N nm Nm the term obtained from the term H by instantiating the degree variables N Nm to the natural numbers n nm respectively and then unfolding the instan tiated term

*De nition Unfolding Unfolding is de ned as follows*

*h u l*

*h u N l*

*def*

*l*

*def*

*h h u N l u*

*As an example f N g a N f f f g a*

*De nition Recurrence domain The recurrence domain of a term H is the set H obtained by unfolding H for all possible values of its degree variables The elements of H are called the constituents of H*

*Note that di erent terms may have the same recurrence domain For in stance the two terms f f N a and f N f a have the same* recurrence domain namely ff a f f a f f f a g

*Uni cation and matching*

*The uni cation of terms also called uni cation di ers from uni cation* of terms in that uni cations are performed on sets of terms instead of terms A substitution is a pair of substitutions whose domains are the rst order variables and the degree variables respectively Two terms have a most gen eral uni er but it may not be unique In an algorithm which computes a nite and complete set of uni ers is given Informally f m g is a complete set of uni ers of two terms L and R if m cover all the most general uni ers of l and r for all pairs l r L R It is shown in

*that computing the intersection of two recurrence domains is the same as*

*nding a complete set of uni ers between two terms whose degree variables* are disjoint

*In an algorithm is given to check that a term T subsumes a term S The subsumption problem is reduced to the uni cation problem by replacing* all rst order variables in S by new constants and then applying a uni cation algorithm The subsumption of terms is also addressed in and an algo rithm is given for the particular case where one of the two terms is a rst order term As it can be seen above we have that if a term T subsumes a term S each term in S is an instance of a term in T However the converse does not hold Let us consider the terms T f N f a and S f f M a we have that S T but there is no substitution such that T S or S T This is in contrast with the

*rst order case where a term t subsumes a term s i s is an instance of t*

*The inclusion of terms is also a decidable problem A simple algorithm is* given in in the case where S and T do not contain any rst order variables

*Recurrence terms in MDGs*

*Intuition*

*By introducing terms in MDGs we intend to generalize a state by an in nite* set of states whose state variables have values that share structural similarities For instance let us consider again the microprocessor of Section After the k th instruction has been executed the machine is in a state represented by the equation c s s s zero where s is iterated k times We may general ize the set of states visited along the sequence path to c by replacing the value of c by the in nite set of termsfzero s zero s s zero s s s zero g rep resented nitely by the term s N zero

*Syntax and semantics*

*We extend the de nition of DF to that of DF by allowing occurrences of*

*terms in the right hand side of the equations In addition to the primary and* secondary variables we consider a third type of variables namely the degree variables that occur in the terms of a DF and which range over natural

*numbers Let P be a DF of type U V such that P fP P P g Then for a given interpretation P represents the set*

*Set P f j j i U P g*

*V i*

*Note that di ers from in Section by the additional quanti cation* over the degree variables occurring in P Intuitively it seems natural that the degree variables are existentially quanti ed since they can be seen as a particular case of secondary variables ranging over natural numbers

*A generator h u N l can be viewed as an ordinary MDG term where is* a constant symbol of concrete sort and an abstract function symbol of arity

*of abstract sort the sort of the elements in its recurrence domain We assume* that those elements are all of the same sort Since a term can occur only in the right hand side of an equation neither N nor participates in the custom symbol order One can verify that the well formedness conditions of MDGs

*can be extended to terms in a straightforward way By considering*

*the terms occurring in a MDG as ordinary MDG terms we can directly* apply to MDGs the basic operations de ned for ordinary MDGs However in the case of the Pruning By Subsumption operation we have to extend the ordinary subsumption so as to be able to recognize that the states represented by the formula x s s N zero are subsumed by those represented by the formula x s N zero This is discussed in the rest of this section

*State exploration for MDGs*

*We have seen in Section that the set of new reachable states in MDG* is based on the PbyS algorithm which applies paths subsumption to remove redundant states Technically to check whether a path i subsumes a path

*it looks for a sub path obtained from by removing some node*

*j j j*

*edge pairs such that i subsumes Let us consider again the example*

*j*

*given at the beginning of this section where after generalization we get the*

*MDG sk c s N zero Now the states reachable from sk after one* transition step are represented by the MDG sk c s s N zero By replacing the ordinary subsumption in PbyS by the subsumption we have that the path sk is subsumed by sk via the substitution N N and therefore the state exploration terminates

*Note that since set inclusion and term subsumption are not equivalent notions* in the algebra of terms the PbyS algorithm based on the subsumption might not be able to detect all states of the frontier that belong to the set of visited states However since we can only detect some particular cases of

*Note that we could easily deal with the case where the base term and the recurrence terms are of di erent sorts by considering the degree variables in terms to range over non null natural numbers This variation is actually used in to avoid the association between unrelated terms*

*Note that the degree variables have to be considered as existential variables and therefore can be renamed in two di erent states*

*non termination we have to evaluate the practical gain of dealing with set* inclusion rather than subsumption This question is out of the scope of the present paper For the examples of Section the subsumption is su cient

*Generalization rule and strategy to solve non termination*

*We propose now a simple generalization technique for introducing terms* during the state exploration The idea is to generalize the value of a state variable as soon as we detect in it a context that is iterated at least twice As an example suppose that we have for a state variable c with the initial value n the following in nite sequence of states produced by applying at each transition the functions f and g alternatively c n c f n c g f n c f g f n c g f g f n c f g f g f n c g f g f g f n

*After the fourth transition the value of c consists of the context g f*

*that is iterated twice Thus we generalize the state c g f g f n by* s c g f N n After two more transition steps we reach the state t g f g f N n and t is subsumed by s via the substitution N N We de ne formally our generalization rule as follows

*De nition Generalization rule Let t be a term X be a variable which does not occur in t and s be a term occurring in t at the position p If s has a proper subterm s which occurs in s at the position q such that s X q subsumes s via the substitution v X then by generalization t is replaced by the term t s q N v p*

*Example Let t be the term g x h f a b f a b g b a y We have s f a b f a b g b a at position p in t s f a b g b a is a proper subterm of s that is subsumed by s U via the substitution g b a U Then t can be generalized by g x h f a b N g b a y*

*The generalization strategy we propose for the state exploration procedure* in MDGs can be described as follows After each transition step the gen eralization rule of De nition is applied to each state of the frontier set whose abstract state variables have values which satisfy the conditions on its application This can be checked automatically To handle the terms in troduced by generalization we extend the existing PbyS algorithm by using the subsumption instead of the ordinary subsumption This strategy will be illustrated on a number of examples in Section The subsumption used in these examples is particularly simple and merely amounts to a one level folding followed by ordinary subsumption This means that few changes are necessary to the existing implementation of the PbyS algorithm in MDG to handle those examples namely the addition of a folding step We do not rule out the fact however that a more complicated subsumption could be needed in some more subtle cases of non termination of ASM state exploration

*Note that the conditions on the application of the generalization rule as de ned* in De nition suggest to generalize a term if it contains a context iterated at least twice This property might be too weak in some cases and might

*yield to inappropriate schematizations Let us consider for instance the in nite sequence n f n f f n g f f n g g f f n f g g f f n f f g g f f n g f f g g f f n produced by a transition re lation which applies in alternance the two functions f and g twice each time Then after the second transition the conditions on the application of the gen eralization rule are satis ed thereby generalizing the term f f n into the*

*term f N n However this term does subsume none of its successor*

*in the sequence*

*A straightforward extension of our generalization rule would be to promote* the number of the iterations of the iterated context into a variable parameter k With k set to the value in the above sequence the rst term on which the extended generalization rule would apply is

*g g f f g g f f g g f f n which would be generalized by T*

*g g f f N n Then after four transition steps from T we would get* g g f f g g f f N n which is subsumed by T via the subsitution N N

# *Examples*

*First example*

*Let us consider a state machine with three state variables c c and c of abstract sort wordn c represents a counter and is incremented via a* functional block represented by the uninterpreted function symbol s which takes x as its input and produces an abstract value s x of abstract sort wordn c resp c is updated via the abstract function f resp g which takes as its abstract input the value of c resp c and produces the abstract output f c resp g c The cross operator eq is of type wordn wordn bool and denotes the equality relation We use the rewriting rule r eq x x Consider now the following transition relation

*Tr eq c c c zero c c c c*

*eq c c c s c c f c c g c*

*where c resp c c is the next state variable of c resp c c Suppose* that the initial state of this machine is s c zero c n c n where n is an abstract variable of sort wordn and zero is a generic constant and let us perform the state exploration using our generalization strategy After one transition we reach the state s and after two transitions we may reach the states s and s as follow

*s c s zero c f n c g n*

*s c zero c f n c f n eq f n g n*

*s c s s zero c f f n c g g n eq f n g n*

*Now state s is subsumed by state s via the substitution f n n and s is generalized by abstracting the values of the state variables c c and c*

*thereby giving the state*

*s c s N zero c f N n c g N n*

*eq f n g n*

*From s we may reach the following states*

*s c zero c f N n c f N n eq f n g n eq f N n g N n*

*s c s s N zero c f f N n c g g N n eq f n g n*

*eq f N n g N n*

*Now state s is subsumed by state s via the substitution f N n n*

*and s*

*is subsumed by s*

*via the substitution fN*

*N N*

*N N*

*N g remember that N N and N are degree variables and therefore* can be renamed in each state Therefore the state exploration terminates after the third transition Note that the procedure also terminates after three transitions if c and c initially contain two arbitrary variables n and n On the other hand the strategy proposed in would suggest to start the state exploration on the generalized initial state

*s c s N zero c f N n c g N n*

*After one step we would reach the following new states*

*s c s s N zero c f f N n*

*c g g N n eq f N n g N n s c zero c f N n c f N n*

*eq f N n g N n*

*The state s is not subsumed by s From s we would reach the following* state

*s c s zero c f f N n c g f N n*

*f N n g N n*

*It is easy to see that the state exploration will generate for c an in nite* sequence of terms of the form g g g g f N n and will never terminate

*Second example*

*Let us consider now a state machine with the state variable y of a concrete* sort with enumeration f g and the state variable x of abstract sort wordn x is updated via an abstract function f which takes x as its input and produces f x of abstract sort wordn The cross term equz x of type wordn bool is used to test if x equals zero r is an input variable of concrete sort with enumeration f g representing the reset We use the following rewriting rule

*r equz zero The transition relation Tr of this machine is de ned as* follows

*Tr r y x zero*

*r y equz x y x f x*

*r y equz x y y x zero*

*r y y y x f x*

*where x resp y is the next state variable of x resp y We start o from* the initial state s x zero y It is easy to see that the state exploration goes into an in nite loop because of the state variable x which produces the in nite sequence of increasing terms zero f zero f f zero f f f zero

*One can also verify that the initial state generalization technique based on*

*variable introduction would also fail to achieve termination on this example Actually if the initial value of x is generalized by an abstract variable n the* state exploration produces the in nite sequence

*x n y x f n y equz n x f f n y equz n*

*x f f f n y equz n since there is no valid substitution such that x f p n y equz n x f q n y equz n with p q*

*The strategy proposed in would suggest to resume the state exploration* with x f N zero as the initial state After one transition we would reach the following states

*s equz f N zero x f f N zero y s equz f N zero x zero y*

*By unfolding these two states and applying the rewriting rule r we get the*

*following new states*

*s x f zero y*

*s equz f f N zero x f f N zero y s equz f f N zero x zero y*

*Now from state s the state exploration would go into the in nite loop*

*equz f f N zero x f f f N zero y*

*equz f f N zero x f f f f N zero y*

*equz f f N zero x f f f f f N zero y*

*since there is no possible substitution for the degree variable N such that*

*equz f f N zero x f n N zero y is subsumed by*

*equz f f N zero x f p N zero y for all p p n A possible solution to achieve termination would be to introduce a second gen eralization step for instance by renaming one of the two occurrences of the* variable N into a new variable N However this would amount to simply drop the constraint equz f f N zero since it would become unrelated to x

*With our strategy the state exploration terminates after the third transition* step Actually from the initial state s we may reach after one transition the states

*s x f zero y*

*s x f f zero y which is generalized into*

*x f N zero y The next state reached from s is*

*s*

*s x f f N zero y which is subsumed by s via the*

*substitution N N and the state exploration terminates*

*Note that the state exploration also terminates if we start from an initial state* where x contains any generic constant n instead of zero Actually we would have the following sequence of states

*s equz n y x f n*

*s equz n y x f f n which after generalization becomes*

*equz n y x f N n and nally*

*s*

*s equz n y x f f N n which is subsumed by s*

*via*

*the substitution N N*

*This means in particular that the state exploration also terminates if we do* not make use of the rewriting rule r

*Third example*

*We borrow here an example from of a state machine which has one state* variable R of concrete sort with enumeration fR R R g The transition relation Tr is as follows

*Tr*

*ie R R R R c c*

*eqz c R R R R c c*

*ie eqz c R R R R c c*

*iy ie R R R R c c*

*iy ie R R R R c inc c*

*iy R R R R c c*

*ie R R R R c c*

*ie R R R R c c*

*where R is the next state variable of R We have the following rewriting rules eqz zero r*

*eqz inc x r*

*The initial state of this machine is s R R c zero Below we* perform the state exploration according to our strategy From s we may

*reach the states s s and s as follows s R R c zero*

*s R R c inc zero*

*s R R c inc inc zero*

*s is generalized by abstracting the value of c by the term inc N zero thereby giving*

*s R R c inc N zero*

*From s we may reach the new state s*

*R R*

*c inc inc N zero*

*which is subsumed by s and therefore the space exploration terminates after*

*the fourth transition*

*In the termination of the space exploration is achieved by starting from* the generalized initial state R R c inc N zero provided a second generalization is performed after the rst state transition Note however that the strategy described in does not address the question of when and how generalization steps must be performed during state ex ploration Moreover in the rst transition step yields the new state

*eqz inc N zero inc N zero By unfolding the constraint*

*eqz inc N zero disappears because the base case gives eqz zero which is rewritten to and the general case gives eqz inc N zero which is rewritten to This simpli cation is possible because we are in* the special case where the cross operator eqz can be fully interpreted on the

*termeqz inc N zero after unfolding However if we modify the ex*

*ample by replacing the cross term eqz c by the cross term even c with the* intended denotation c is an even number then the simpli cation of the constraint even inc N zero by unfolding is only possible in the base case even zero is rewritten to but no longer in the general case since even inc inc N zero cannot be simpli ed One can verify that the strategy used in would fail to achieve termination on the modi ed version of the example while our strategy terminates in this case too Actually the termination of the state exploration above described can be achieved without the use of the rewriting rules r and r

*Conclusions*

*We have presented a generalization method based on term schematization for* solving some cases of non termination in the state exploration procedure of Abstract States Machines represented by MDGs It is based on a general ization rule which aims at replacing a rst order term presenting a certain recurring pattern by a term i e a meta term which represents in a nite way an in nite sequence of terms sharing the same pattern This rule allows us to achieve termination of the state enumeration in the case where non ter mination is caused by a certain repetitive pattern which grows in nitely along the sequence of reachable states Since terms are capable of propagating some information about the structure of the terms they generalize we think

*that they are particularly useful in reachability analysis because they reduce* the risk of producing false negatives This is in contrast with the techniques based on generalization variable introduction which may produce very poor generalizations

*Our method improves over the previous proposals and in particular over where the idea of using term schematization in MDGs has been rst proposed First and foremost it can be applied fully automatically and does not require* the user to provide the state variable to be generalized nor its generalized value Second generalization is not performed at the initial state which allows to cope with situations where a state variable gives rise to di erent patterns of divergence in alternative branches of the state exploration Moreover it allows to exploit some information provided by the axioms which partially interpret abstract functions on the initial values if any Our technique is currently under implementation into the MDG tools and future work will include the study of its applicability to the reachability analysis in real designs We also intend to enhance our generalization rule to detect and handle more complicated cases of recurrent term structures We may consider in particular the method proposed in for inferring recursive patterns based on the notion of embedding relation

*References*

*O A t Mohamed E Cerny and X Song Mdg based Veri cation by Retiming and Combinational Transformations Proccedings of the IEEE th Great Lakes Symposium on VLSI Louisiana USA*

*O A t Mohamed X Song E Cerny S Tahar and Z Zhou Solving the Non Termination of Abstract Implicit State Enumeration Using a Circuit Transformation A Case Study on the Island Tunnel Controller Research report University of Montreal*

*O A t Mohamed X Song and E Cerny On the Non termination of MDG based Abstract State Enumeration Proccedings of the IFIP W Advanced Research Working Conference on Correct Hardware Design and Veri cation Methods Charme Montr eal October*

*IFIP Chapmann Hall*

*A Amaniss M Hermann D Lugiez Set Operations for Recurrent Term Schematization Proccedings of the TAPSOFT Lille*

*France volume of LNCS pages Springer Verlag*

*A Amaniss M ethodes de sch ematisation pour la d emonstration automatique PhD thesis Universit e Henri Poincar e Nancy*

*K D Anon N Boulerice E Cerny F Corella M Langevin X Song S Tahar Y Xu Z Zhou MDG tools for the Veri cation of RTL Designs In Proceedings of the Conference on Computer Aided Veri cation CAV New Jersey USA July*

*R E Bryant Graph based Algorithms for Boolean Function Manipulation IEEE Transactions on Computers August*

*E Cerny F Corella M Langevin X Song S Tahar and Z Zhou Automated Veri cation with Abstract State Machines Using Multiway Decision Graphs In Formal Hardware Veri cation Methods and Systems in Comparison volume of LNCS pages Springer Verlag*

*H Chen J Hsiang and H C Kong On nite Representations of In nite Sequences of Terms Proceedings of the nd International Workshop on Conditional and Typed Rewriting Systems Montreal*

*Canada volume of LNCS pages Springer Verlag*

*H Chen and J Hsiang Recurrence domains Their Uni cation and Application to Logic Programming Information and Computation*

*H Comon On Uni cation of terms with Integer Exponents Mathematical System Theory*

*F Corella Z Zhou X Song M Langevin and E Cerny Multiway Decision Graphs for Automated Hardware Veri cation Formal Methods in System Design*

*F Corella Z Zhou X Song M Langevin and E Cerny MDG Algorithms I Technical Report University of Montreal*

*M Hermann On the Relation Between Primitive Recursion Schematization and Divergence Proceedings of the rd Conference on Algebraic and Logic Programming Volterra Italy volume*

*of LNCS pages Springer Verlag*

*M Hermann and G Salzer On the Word Subsumption and Complement Problem for recurrent Term Schematizations In L Brim J Gruska and J Zlatuska editors Proceedings of the rd International Symposium on Mathematical Foundations of Computer Science MFCS Brno Czech Republic volume of LNCS pages Springer Verlag*

*newblock The Theory of Well quasi ordering A Frequently Discovered Concept Journal of Combinatorial Theory November Ser A*

*G Salzer The Uni cation of In nite Sets of Terms and Its Applications In Voronkov A editor Proceedings of the rd International Conference on Logic Programming and Automated Reasoning St Petersburg Russia volume of LNCS pages*

*Springer Verlag*

*S Tahar Z Zhou X Song E Cerny and M Langevin Formal Veri cation of an ATM Switch Fabric using Multiway Decision Graphs In Proccedings of the Great Lakes Symposium on VLSI GLS VLSI Arres Iowa USA March IEEE Computer Society Press*

*Y Xu E Cerny X Song F Corella and O A t Mohamed Model Checking for A rst Order Temporal Logic using Multiway Decision Graphs In Alan J H and Moshe Y Vardi editors Proceedings th International Conference on Computer Aided Veri cation CAV Vancouver BC Canada volume of LNCS pages Springer Verlag*

*Z Zhou MDG tools developer s manual Technical Report University of Montreal*

*Z Zhou X Song F Corella M Langevin and E Cerny MDG Algorithms II Technical Report University of Montreal*

*Z Zhou X Song S Tahar E Cerny F Corella and M Langevin Formal Veri cation of the Island Tunnel Controller using Multiway Decision Graphs In Proceedings of the International Conference on Formal Methods in Computer Aided Design FMCAD USA pages*